

Root Cause Analysis

October 15, 2008

LACSS Resilience Workshop

Presented by Jon Stearley

Sandia National Laboratories



Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.





Agenda

(10 slides, 3 goals)

Get your help on Root Cause

- How to represent interdependencies?

Encourage you towards standardized validation data

- Component Operations Status (COS)

Make you aware of some other work

- Sisyphus (Logs)
- 9Lives (OS)

Context: Resilience Activities at Sandia

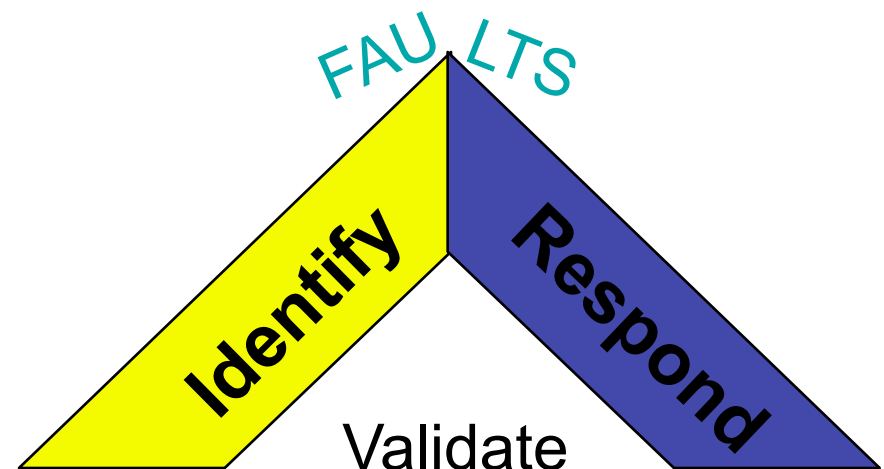
Goal: Automatic *identification* and *response* to faults

Identify Faults (CSSE)

- Failure Prediction
- Root cause analysis
- *Impact: enable timely and focused response*

Respond (LDRD):

- System-Directed Resilience
- *Impact: enable apps to run continuously despite faults*



Root Cause: Big Cheese Model

* Holes: incomplete data, incomplete time, incomplete components, incomplete dependencies

Given hints (influences search path):

Distinguishing Symptoms

- Text (logs, username, job id, rank id, ...)
- Numbers (temperature, correlation, ...)

Suspect Components

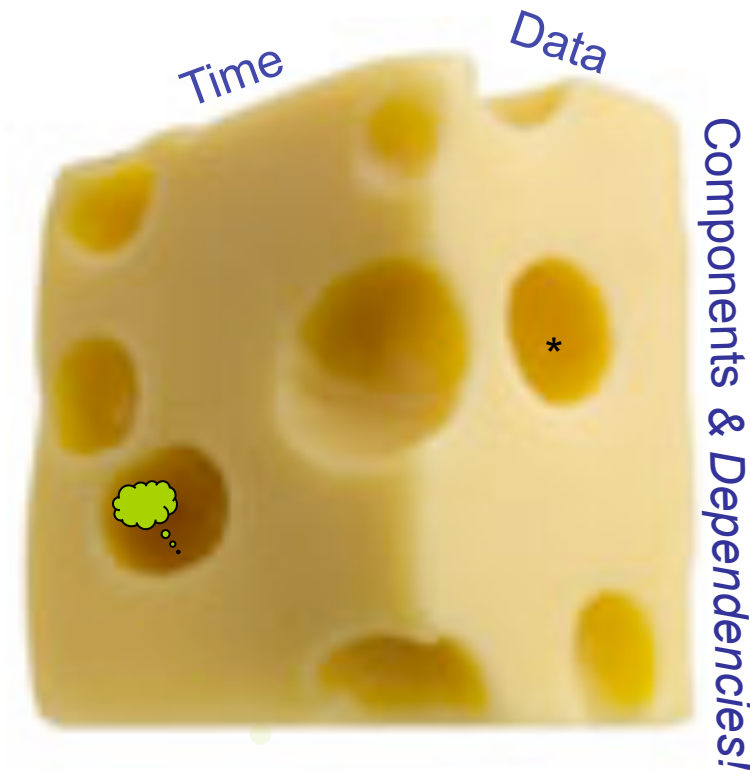
- Hardware (racks, cores, routes, ...)
- Software (daemons, apps, libraries, ...)

Dependencies

- Functional, Physical
- Static, Dynamic

Identify:

Root Cause (likelihood ranked)



What Stinks?!#

Root Cause: Big Cheese Model

* Holes: incomplete data, incomplete time, incomplete components, incomplete dependencies

Given hints (influences search path):

LABELS

Distinguishing Symptoms

- Text (logs, username, job id, rank id, ...)
- Numbers (temperature, correlation, ...)

VERTICES

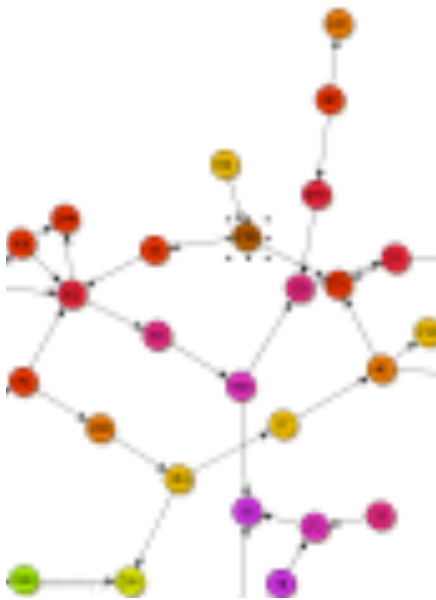
Suspect Components

- Hardware (racks, cores, routes, ...)
- Software (daemons, apps, libraries, ...)

EDGES

Dependencies

- Functional, Physical
- Static, Dynamic



$G=(V,E,L)$

Properties?

Useful model?

Useful algorithms?

Identify:

Root Cause (likelihood ranked)

Root Cause: Flakey Link (e.g.)

User u1, jobid 1, node X, ...
“timeout writing...”

(network routes)

node Z

User u1, jobid 2 filesystem f, node Y, ...
“CRC error Y->Z”

Link L
retransmit=5

I/O server s

filesystem f

Disk d

Input: “jobid1, jobid2”

Output: “link L, retransmit=5”

Input: “CRC error, today”

Output: “link L, user u1, jobid 1”



FY'09 Root Cause Deliverables

First Quarter

- Mathematical formulation of at least three important but currently non-computable root cause scenarios based on current systems

Second Quarter

- Select appropriate algorithms for solution.

Third Quarter

- Demonstrate proof-of-concept solutions.

Fourth Quarter

- SAND report

Validation:

“Quantify our ability to predict node failures on the TLCC platform”

Component Operations Status (COS)

Production Uptime (PU) = ready for immediate use by one or more production user

Scheduled Downtime (SD) = not in PU for scheduled reasons

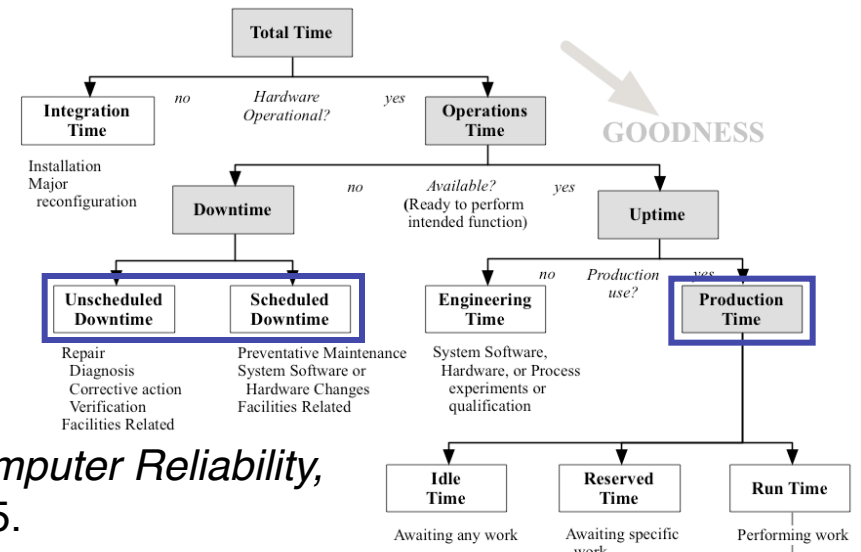
Unscheduled Downtime (UD) = not in PU for unscheduled reasons

FAILURE = the onset of Unscheduled Downtime

FY09:

1. Collect per-node COS on TLCC (admins express “SD” on CLI)
2. Perform prediction experiments on symptomatic data (numeric, text)
3. Use COS to quantify (and validate) prediction results

COS is a proposed standard

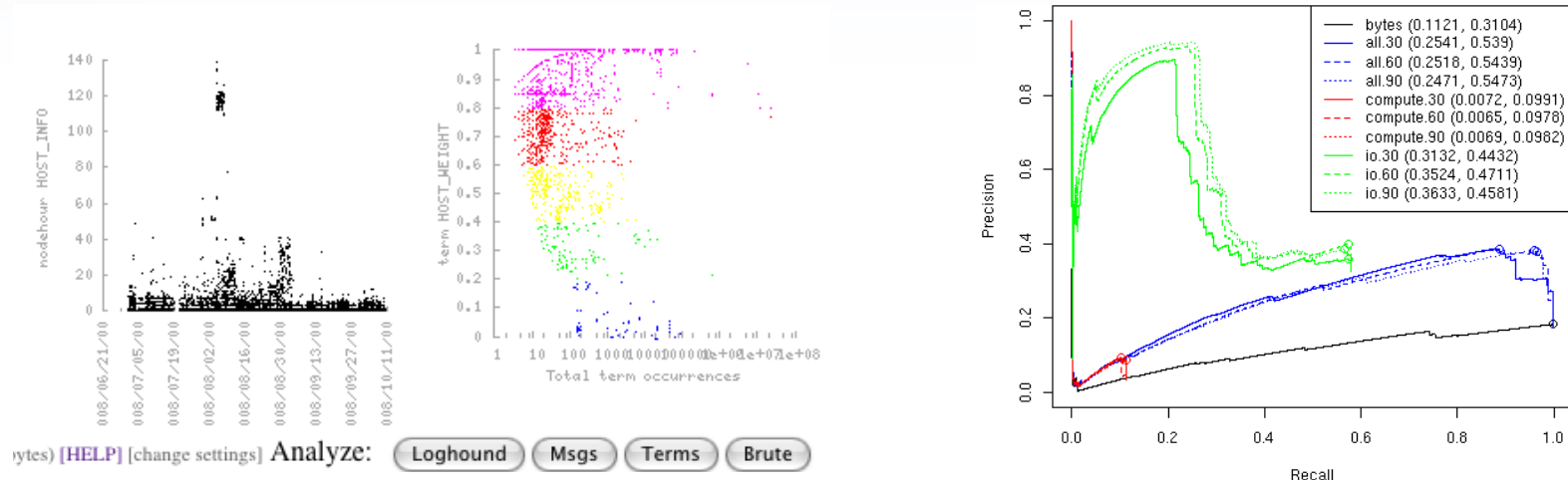




Oliner, Stearley. *Alert Detection in System Logs*, ICDM'08.

www.cs.sandia.gov/sisyphus

cfdrr.usenix.org/data.html#hpc4



```
Aug 4 14:00:12 10.1.0.84 local7 info DMT_EMT Unable to read disk 5S block 0x0002ED32 - Adding to Reassign List.
Aug 4 14:00:14 10.1.0.84 local7 info DMT_EMT Unable to read disk 5S block 0x0002ED56 - Adding to Reassign List.
Aug 4 14:00:15 10.1.0.84 local7 info DMT_EMT Unable to repair disk address 191794 LUN 8, 00000176 DLR:0, DLG:1, DRR:0, DEL:0, DELR:0
Aug 4 14:00:15 10.1.0.84 local7 info DMT_EMT Unable to repair disk address 191830 LUN 8, 00000176 DLR:0, DLG:1, DRR:0, DEL:0, DELR:0
Aug 4 14:00:15 10.1.0.84 local7 info DMT_EMT Unable to recover defects on disk 5S LUN 8, 00000176 DLR:0, DLG:1, DRR:0, DEL:0, DELR:0
Aug 4 14:00:15 10.1.0.84 local7 info DMT_EMT DMT EMT invalidate node LUN 8, 00000176 DLR:0, DLG:1, DRR:0, DEL:0, DELR:0, DERR:0
Aug 4 14:00:15 10.1.0.84 local7 crit DMT_EMT Warning: EMT multi-channel failure detection: BBM disabled: disk 5S, LUN 8, 00000176 DLR:0, DLG:1, DRR:0, DEL:0, DELR:0, DERR:0 r0 w0 i1 f0 fr2 ea:0,10
Aug 4 14:00:15 10.1.0.84 local7 info DMT_EMT Reload Node tier: 5 LUN 8, 00000176 DLR:0, DLG:1, DRR:0, DEL:0, DELR:0, DERR:0 r0 w0 i1 f0 fr2 ea:0,10
Aug 4 14:00:17 10.1.0.84 local7 debug Disk_Int DISKS3: Received Rsp for incomplete exchange 00000021 Completed:00026400 Exp:00040000
Aug 4 14:00:17 10.1.0.84 local7 debug Disk_Int DISKS2: StartCmdRetry - Response for Drive 0x05S exch:00000021
Aug 4 14:00:17 10.1.0.84 local7 debug Disk_Int DISKS4: Cmd:28 LBA:0002EC00 Length:00040000
Aug 4 14:00:17 10.1.0.84 local7 debug Disk_Int DISKS4: Current Error SCSI Status:00000B02
Aug 4 14:00:17 10.1.0.84 local7 debug Disk_Int DISKS4: Under-run Residual Count:00019C00
Aug 4 14:00:17 10.1.0.84 local7 debug Disk_Int DISKS4: Response DL:00000008 Response:00000000 00000000
Aug 4 14:00:17 10.1.0.84 local7 debug Disk_Int DISKS4: Sense Key:03 Info:0002ED32
Aug 4 14:00:17 10.1.0.84 local7 debug Disk_Int DISKS4: ASC:11 ASCQ:00 FRU:E4
```



Responding to Faults (9Lives) System-Directed Resilience (LDRD)

Three Primary Research Thrusts

- **Application quiescence**
 - Suspend App, handle in-flight messages and in-progress IO
- **Efficient state management**
 - Identify critical state (app characterization, app guidance, changesets)
 - System guidance for when to extract state (eg MTTI)
 - Explore diskless methods and compression
- **Fault recovery**
 - System software to support dynamic node allocation
 - Explore network virtualization to abstract physical node ID from software.
 - Explore efficient methods for state recovery (roll-back, roll-forward techniques)

Low Overhead (easier to develop and support, faster to run)

- **Lightweight Kernel (Kitten)**
- **Stateless Networking Protocol (Portals)**



End

Get your help on Root Cause

- How to represent interdependencies?

Convince you towards standardized validation metrics

- Component Operations Status (COS)

Make you aware of some other work

- Sisyphus (Logs)
- 9Lives (OS)

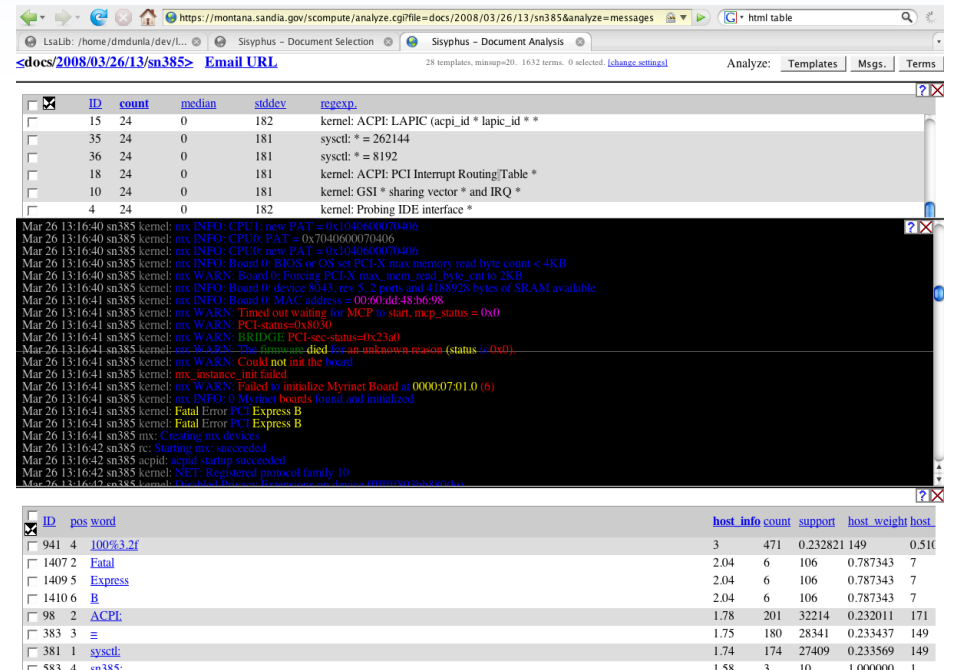
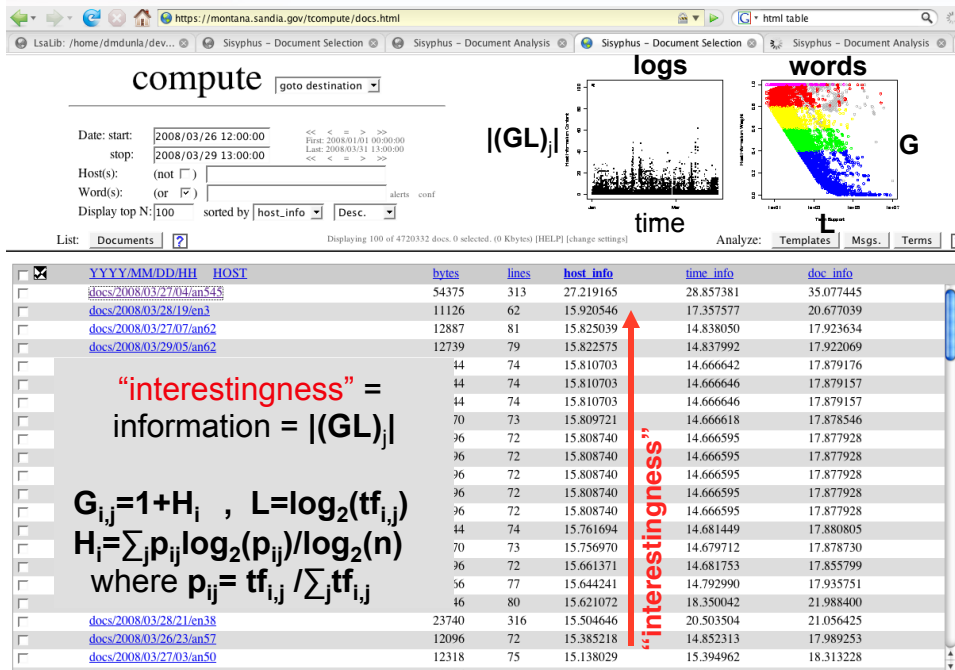
Extra Slides

Sisyphus

Automatic Fault Detection



Jon Stearley
jrstear@sandia.gov



1. Automatically rank logs by information content.

2. Automatically color words by information weight.

3. Automatically deduce word and message patterns.

Similar computers correctly performing similar work should produce similar logs (anomalies warrant investigation).





Identifying and Predicting Faults

Research to Identify Root Cause

- Develop mathematical methods to describe and track dependencies between system components
- Develop tools to analyze models, identify root cause, and provide feedback to fault-response systems (e.g., the system directed LDRD)

Research on Fault Prediction

- Develop methodologies for anomaly detection and quantification
- Correlate anomalous behaviors with root causes
- Use automated anomaly detection and learned correlations to predict failures with a calculated level of confidence

Impact

- Critical enabler for automated response (also useful to admins)
- Mathematics necessary to design and operate truly resilient systems
- Quantify the system-wide impact of failures.



FY'09 9Lives Deliverables

First Quarter

- Enumerate list of faults, likelihood of fault, and possible response (Stearley)
- Investigate options for quiescence (Brightwell)
- Investigate diskless state management and node recovery (Oldfield)
- Investigate related projects – BLCR (Kordenbrock)

Second Quarter

- Complete design of code to trace application memory usage (Pedretti)
- Complete design of network virtualization layer (Brightwell)
- Complete design of fake RAS system (Riesen)
- Complete design of diskless state management and recovery (Oldfield)

Fourth Quarter

- Memory-use characterizations for selected applications (Pedretti)
- Demonstrate response to link failures – may not need node recovery (Laros)
- Demonstrate network virtualization layer (Brightwell)
- Demonstrate fake RAS – inject faults (Riesen)

FY'09 Prediction Deliverables

- **First Quarter**
 - Selection of algorithms for anomaly detection (multivariate distributions and time series phenomena)
 - Define database schema for collection of system variables and for representation of data to support the different analyses
- **Second Quarter**
 - Quantification methodologies, failure definition and recording mechanisms, data collection (CPM, Log, Workload, COS, etc.)
- **Third Quarter**
 - Implementation and validation of anomaly detection algorithms
 - Selection of algorithms for correlation of failures with anomalies (classification and root cause inference)
 - Attributes selected for predictive analysis
 - Scalable data collection and representation
- **Fourth Quarter**
 - Implementation and validation of failure to anomaly correlation techniques
 - Write SAND report documenting:
 - Quantification of the ability to predict failures on the TLCC platform
 - Additional information/data that should be collected, as well as suggested scalable collection mechanisms, in order to improve failure predictability on future platforms

Proposed Process for Collecting per-node COS on TLCC

**Scheduled
Downtime**

**Production
Uptime**

**Unscheduled
Downtime**

SLURM and MOAB logs and databases are postprocessed into COS records.

All downtimes are counted as Unscheduled, unless distinguishable otherwise:

Only the admins know if a downtime is scheduled or not, so they must be involved in distinguishing them. The easiest way to do this would be to use a flag to scontrol and mrsvctl like:

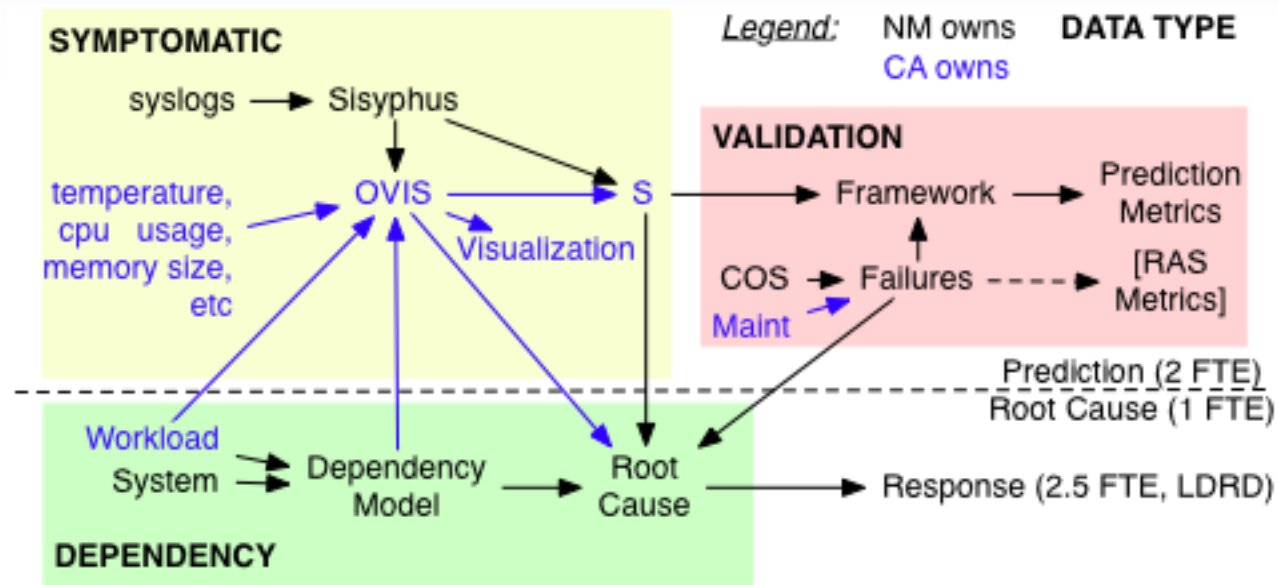
```
scontrol -sd [cause...]  
mrsvctl -sd [cause...]
```

Which would be shorthand for

```
scontrol reason="SD: [cause]"  
mrsvctl -a ACCT=SD -D "SD: [cause]"
```

Where [cause] is an optional argument describing the downtime.
-sd could be added, aliased, or wrapped. More details are on the next slide.

SNL Resilience Interactions



Details:

Prediction Signal $S: s_i=(t,n,c)$ where t is time, n is node, and c is confidence

Failures $F: f_j=(t_f,n,d)$ where t_f is time of failure, n is node, and d is duration

COS is Component Operations Status (uptime, sched and unsched downtime)

Maint is maintenance records (hardware replacements etc)

Framework: how well does S predict F ? $P(F | S)$

Prediction Metrics: Precision, Recall, AUC

Workload: jobs (user, nodes, duration,...) and executables (versions, libraries,...)

System: hardware and software configurations (cores, cables, routes, daemons,...)

Dependency Model: $G=(V,E)$ where V are components and E are dependencies

Root Cause: $P(G' | S,F)$ based primarily on dependency model

*Root Cause: $P(n | S)$ based primarily on symptomatic inference

[RAS Metrics]: no FY09 work is planned, but FYI COS is also key for:

MTTI, MTTR, component pareto (including CCF via Dependency Model), etc

Root Cause: Graph Model

* Holes: incomplete data, incomplete time, incomplete components, incomplete dependencies

Given hints (influences search path):

LABELS

Distinguishing Symptoms

- Text (logs, username, job id, rank id, ...)
- Thresholds (temperature limits, ...)
- Waveforms (cpu profile, ...)
- Correlations

VERTICES

Suspect Components

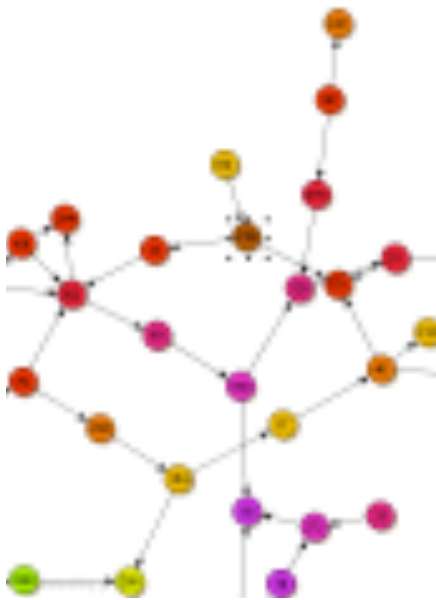
(paths)

- Hardware (racks, cores, routes, ...)
- Software (daemons, apps, libraries, ...)

EDGES

Dependencies

- Functional, Physical
- Static, Dynamic



$G=(V,E,L)$

Properties?

Useful model?

Useful algorithms?

Identify:

Root Cause (likelihood ranked)